II International Conference on Particle-based Methods - Fundamentals and Applications
PARTICLES 2011
E. Oñate and D.R.J. Owen (Eds)

COUPLING OF GRANULAR MEDIA AND FLUID FLOW SOLVED BY THE FINITE POINTSET METHOD

Jan Marburger* **and Jörg Kuhnert***

*Fraunhofer Institute for Industrial Mathematics
Fraunhofer-Platz 1
67663 Kaiserslautern, Germany
e-mail: {marburger,kuhnert}@itwm.fhg.de

Key words: Finite Pointset Method, Granular Materials, Fluid Flow

Abstract. In this paper we present strategies of simulating particles with mass like stones or dust in flow problems. The focus will be on the modelling of the particles, e.g. particle-particle and particle-wall collisions, and the coupling of the fluid phase to the particles. The latter means the interpolation of the fluid or gas phase data, provided by the finite pointset method, and the granular phase. Note that there is a large difference in density of the point set needed for the fluid phase and the one describing the granular phase. Therefore, an efficient interpolation algorithm is shown in order to decrease the numerical effort. Moreover, the modelling of an inner energy, modelling stress in the particles, is described which also allows a breakage of particles if the stress becomes to large.

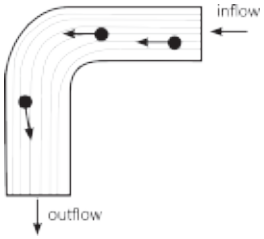
1 Introduction

In many industrial processes the simulation of granular media is of great interest. For example crushing of rocks, which can be considered as particles in a gas flow or the mixing of concrete, which yields particles in a fluid flow, cf. figure 1. Very often the fluid flow involves also free boundaries, for example figure 1(a), or moving domains, like for mixing applications. Especially these problems can be solved by particle methods, like the finite pointset method (FPM), very efficiently. For this reason, we investigate the coupling of a particle phase with a fluid phase solved by FPM.

First, we state the model used for the particle phase, that is, particle-particle, particle-wall and particle-fluid interaction. Second we introduce the idea of the finite pointset method and the numerical implementation of the fluid phase.

Then the coupling is investigated. In particular we show the algorithm for mapping the fluid data to the particle phase and state the numerical implementation. We also introduce a model for an inner energy based on the total kinetic energy which allows the

(a) Stones at a coast driven by water waves.



(b) Particles driven by a fluid flow in a bent pipe.

Figure 1: Examples for particles with mass in flows.

consideration of internal stresses of the particles. If this stress becomes too large, the particle breaks.

Finally, the derived model is applied to a 2D particle separator.

2 Model for the Granular Phase

In this section we introduce models for the granular phase which are mainly based on [2]. First we consider particle-particle interactions followed by particle-wall interaction. Finally, the force of the fluid to the particles is taken into account. For reasons of simplicity we assume the particles to be spheres.

2.1 Particle-Particle Interaction

In this paper we use a hard sphere model, cf. [2, p.129], which is based on the integrated Newtonian equations. In the following, we denote the pre-collision quantities by $\bar{\cdot}$, e.g. the pre-collision velocity is denoted by $\bar{\mathbf{v}}_1$ and the post-collision one by $\mathbf{v}_1$. The contact phase is illustrated in figure 3. For spheres with homogeneous density we obtain the conditions

$$\begin{aligned} m_1(\mathbf{v}_1 - \bar{\mathbf{v}}_1) &= \mathbf{J} \\ m_2(\mathbf{v}_2 - \bar{\mathbf{v}}_2) &= \mathbf{J} \end{aligned} \qquad \text{and} \qquad \begin{aligned} I_1(\boldsymbol{\omega}_1 - \bar{\boldsymbol{\omega}}_1) &= r_1\mathbf{n} \times \mathbf{J} \\ I_2(\boldsymbol{\omega}_2 - \bar{\boldsymbol{\omega}}_2) &= r_2\mathbf{n} \times \mathbf{J} \end{aligned}$$

where $\mathbf{v}$ denotes the velocity of the particle, $\boldsymbol{\omega}$ the angular velocity, $\mathbf{J}$ the impulsive force and $\mathbf{n}$ the normal vector pointing from particle 1 to 2. The material parameters are the radius r, the mass m and the moment of inertia I.

We define the relative velocity of the particles and at the contact point by

$$\mathbf{G} := \bar{\mathbf{v}}_1 - \bar{\mathbf{v}}_2 \qquad \text{and} \qquad \mathbf{G}_c := \mathbf{G} + r_1\bar{\boldsymbol{\omega}}_1 \times \mathbf{n} + r_2\bar{\boldsymbol{\omega}}_2 \times \mathbf{n}$$

respectively. The tangential velocity and tangential vector is then given by

$$\mathbf{G}_{ct} = \mathbf{G}_c - (\mathbf{G} \cdot \mathbf{n})\mathbf{n} \qquad \text{and} \qquad \mathbf{t} = \frac{\mathbf{G}_{ct}}{|\mathbf{G}_{ct}|}.$$

During the contact the particles can slide the whole contact time or stop sliding during the collision. These two cases are distinguished by the relation

$$\frac{\mathbf{G}\cdot\mathbf{n}}{|\mathbf{G}_{ct}|} < \frac{2}{7(1+e)f} \tag{1}$$

where e is the restitution coefficient and f the friction coefficient, which are given material parameters. If condition (1) is satisfied, the particles slide during the contact and we obtain the post collision quantities

$$\begin{aligned}
\mathbf{v}_1 &= \bar{\mathbf{v}}_1 - (\mathbf{n} - f\mathbf{t})(\mathbf{G}\cdot\mathbf{n})(1+e)\frac{m_2}{m_1+m_2}\\
\mathbf{v}_2 &= \bar{\mathbf{v}}_2 + (\mathbf{n} - f\mathbf{t})(\mathbf{G}\cdot\mathbf{n})(1+e)\frac{m_1}{m_1+m_2}\\
\boldsymbol{\omega}_1 &= \bar{\boldsymbol{\omega}}_1 - \frac{5}{2r_1}(\mathbf{G}\cdot\mathbf{n})(\mathbf{n}\times\mathbf{t})(1+e)f\frac{m_2}{m_1+m_2}\\
\boldsymbol{\omega}_2 &= \bar{\boldsymbol{\omega}}_2 - \frac{5}{2r_2}(\mathbf{G}\cdot\mathbf{n})(\mathbf{n}\times\mathbf{t})(1+e)f\frac{m_1}{m_1+m_2}.
\end{aligned}$$

If condition (1) is not satisfied, the particles stop sliding during the contact and we obtain

$$\begin{aligned}
\mathbf{v}_1 &= \bar{\mathbf{v}}_1 - \left((\mathbf{G}\cdot\mathbf{n})(1+e)\mathbf{n} + \frac{2}{7}|\mathbf{G}_{ct}|\,\mathbf{t}\right)\frac{m_2}{m_1+m_2}\\
\mathbf{v}_2 &= \bar{\mathbf{v}}_2 + \left((\mathbf{G}\cdot\mathbf{n})(1+e)\mathbf{n} + \frac{2}{7}|\mathbf{G}_{ct}|\,\mathbf{t}\right)\frac{m_1}{m_1+m_2}\\
\boldsymbol{\omega}_1 &= \bar{\boldsymbol{\omega}}_1 - \frac{5}{7r_1}|\mathbf{G}_{ct}|\,(\mathbf{n}\times\mathbf{t})\frac{m_2}{m_1+m_2}\\
\boldsymbol{\omega}_2 &= \bar{\boldsymbol{\omega}}_2 - \frac{5}{7r_2}|\mathbf{G}_{ct}|\,(\mathbf{n}\times\mathbf{t})\frac{m_1}{m_1+m_2}
\end{aligned}$$

For further details we refer to [2].

2.2 Particle-Wall Interaction

The particle-wall interaction is more complex than the particle-particle interaction. On the one hand we have to distinguish the sliding and non-sliding cases, similar to the above case. On the other hand, the handling also depends on the geometry. First, we consider the geometrical properties. In three dimensions a ball can hit either a surface, edge or corner. If a ball hits a corner point or an edge then we set the normal vector $\mathbf{n}$ according to figure 2 and use the particle-particle approach shown above. In particular, particle 1 is the approaching one and particle 2 is a ghost particle at the contact point. This ghost particle models the wall by setting $r_2 = 0, \mathbf{v}_2 = 0, \boldsymbol{\omega}_2 = 0$ and $m_2 = \infty$, i.e. very large.

In the case that a particle hits a surface we define the normal and tangential directions by

$$N = \mathbf{n}, \qquad T = \mathbf{v} - (\mathbf{v}\cdot\mathbf{n})\mathbf{n} \qquad \text{and} \qquad S = N \times T$$

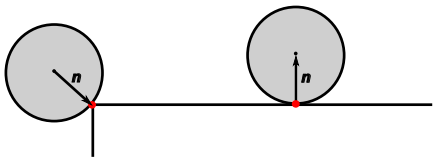


Figure 2: Particle-Wall interaction.

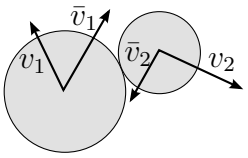


Figure 3: Particle-Particle collision.

and denote by $v_T := \mathbf{v} \cdot T/|T|$ the quantity in the corresponding direction. Similar to the particle-particle interaction, we have to distinguish the sliding and non-sliding case which yields the following post-collision quantities. If

$$\frac{v_N}{|\mathbf{v}|} < -\frac{2}{7f(e+1)} \tag{2}$$

is satisfied we obtain

$$\begin{aligned} \mathbf{v}_T &= \frac{5}{7}\left(\bar{\mathbf{v}}_T - \frac{2r}{5}\bar{\boldsymbol{\omega}}_S\right) & & \boldsymbol{\omega}_T = \frac{\mathbf{v}_S}{r} \\ \mathbf{v}_N &= -e\bar{\mathbf{v}}_N & \text{and} \qquad & \boldsymbol{\omega}_N = \bar{\boldsymbol{\omega}}_N \\ \mathbf{v}_S &= \frac{5}{7}\left(\bar{\mathbf{v}}_S + \frac{2r}{5}\bar{\boldsymbol{\omega}}_T\right) & & \boldsymbol{\omega}_S = -\frac{\mathbf{v}_T}{r} \end{aligned}$$

otherwise

$$\begin{aligned} v_T &= \bar{\mathbf{v}}_T + f(e+1)\bar{\mathbf{v}}_N & & \boldsymbol{\omega}_T = \bar{\boldsymbol{\omega}}_T \\ v_N &= -e\bar{\mathbf{v}}_N & \text{and} \qquad & \boldsymbol{\omega}_N = \bar{\boldsymbol{\omega}}_N \\ v_S &= \bar{\mathbf{v}}_S & & \boldsymbol{\omega}_S = \bar{\boldsymbol{\omega}}_S + \frac{5}{2r}f(e+1)\bar{\mathbf{v}}_N. \end{aligned}$$

Again we refer to [2] for more details.

2.3 Particle-Fluid Interaction

In this paper we only consider the influence of the fluid and the surrounding to the particles, not vice versa. First, we consider the gravity $\mathbf{g}$. The force on the particle is given by $\mathbf{F}_{part} = m\mathbf{g}$. This yields, by neglecting collisions, the relation $\dot{\mathbf{v}}_{part} = \mathbf{g}$. Similarly, the force induced by the fluid is handled. Here we obtain $\mathbf{F}_{part} = \alpha(\mathbf{v}_{fluid} - \mathbf{v}_{part})$, where α is a material constant depending on the properties of the fluid and the particle. In a similar manner other forces, e.g. the Magnus force, can be modelled.

3 Model for the Continuous Phase

In this section we give a short introduction to the finite pointset method (FPM) and the numerical implementation of the Navier-Stokes equations.

3.1 Finite Pointset Method

The basic idea of this method is exemplified by the Laplacian. Let $\Omega \subset \mathbb{R}^2$ be a bounded domain and $f : \Omega \to \mathbb{R}$ a sufficiently smooth function. Moreover, let $P = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, $\mathbf{x}_i = (x_i, y_i) \in \Omega$ denote a given point set. Then we approximate f by $f_h(\mathbf{x}) = \sum_{j=1}^{N} c_j(\mathbf{x}) f_j$ where c_j are approximation weights and $f_j = f(\mathbf{x}_j)$ supporting values. For the Laplacian we get

$$\Delta f(\mathbf{x}) \simeq \Delta\Big(\sum_{j=1}^{N} c_j(\mathbf{x}) f_j\Big) = \sum_{j=1}^{N} (\Delta c_j(\mathbf{x})) f_j =: \sum_{j=1}^{N} \tilde{c}_j(\mathbf{x}) f_j.$$

To obtain the weights $\tilde{c}_j$ we use the following properties of the continuous Laplacian

$$\Delta\text{const} = \Delta x = \Delta y = \Delta(xy) = 0 \qquad \text{and} \qquad \Delta(x^2) = \Delta(y^2) = 2.$$

Hence, for each point $\mathbf{x}_j \in P$ the weights $\tilde{c}_j(\mathbf{x})$ have to satisfy

$$\sum_{j=1}^{N} \omega_j(\mathbf{x})\tilde{c}_j(\mathbf{x}) = 0, \quad \sum_{j=1}^{N} \omega_j(\mathbf{x})\tilde{c}_j(\mathbf{x})(x - x_j)(y - y_j) = 0, \quad \sum_{j=1}^{N} \omega_j(\mathbf{x})\tilde{c}_j(\mathbf{x})(x - x_j) = 0$$

$$\sum_{j=1}^{N} \omega_j(\mathbf{x})\tilde{c}_j(\mathbf{x})(y - y_j) = 0, \quad \sum_{j=1}^{N} \omega_j(\mathbf{x})\tilde{c}_j(\mathbf{x})(x - x_j)^2 = 2, \quad \sum_{j=1}^{N} \omega_j(\mathbf{x})\tilde{c}_j(\mathbf{x})(y - y_j)^2 = 2$$

in a neighbourhood of $\mathbf{x}$, which is defined by the smoothing length (cf. figure 4), and the weighting functions $\omega_j(\mathbf{x})$ depending on the distance from $\mathbf{x}$ to $\mathbf{x}_j$, e.g. a Gaussian function as shown in figure 5. This finally yields an underdetermined linear system, as we use more supporting points than approximation conditions. The resulting system is solved by a QR factorisation, for instance.

In this fashion, all spatial derivatives are approximated. Also complex boundary conditions can be implemented in that way. For example, the derivative in normal direction, i.e. $\nabla f \cdot \mathbf{n}$, can be approximated by the conditions

$$\nabla\text{const} \cdot \mathbf{n} = 0, \qquad \nabla x \cdot \mathbf{n} = n_x \qquad \text{and} \qquad \nabla y \cdot \mathbf{n} = n_y$$

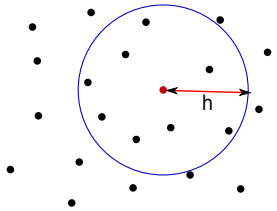


Figure 4: Point set and smoothing length h.

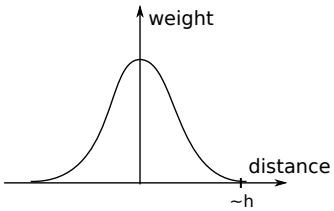


Figure 5: Weight function.

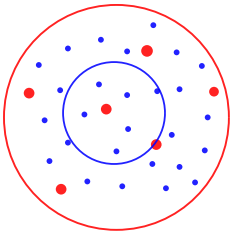


Figure 6: Illustration of h_{fluid} and h_{part}.

for $\mathbf{n} = (n_x, n_y)^T$. Moreover, the extension of the above approach to 3D is straight forward, i.e. appropriate conditions for the z-direction are added. For more details we refer to [3, 4].

3.2 Numerical Implementation of the Navier-Stokes Equation

The fluid phase is modelled by the incompressible Navier-Stokes equation which is roughly given by

$$\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \Delta \mathbf{u} = -\nabla p \qquad \text{in } \Omega \times (0,T) \tag{3a}$$

$$\operatorname{div} \mathbf{u} = 0 \qquad \text{in } \Omega \times (0,T) \tag{3b}$$

$$\mathbf{u} = \mathbf{g} \qquad \text{on } \Gamma \times (0,T) \tag{3c}$$

where $\mathbf{g}$ satisfies the compatibility condition $\int_\Gamma \mathbf{g} \cdot \mathbf{n} \, d\omega(x) = 0$. Moreover, feasible initial conditions are given. We decouple the velocity $\mathbf{u}$ and pressure p in the momentum equation (3a) and discretise it in time, i.e.

$$\begin{aligned} \frac{1}{\tau}(\mathbf{u}^* - \mathbf{u}^n) + \mathbf{u}^* \cdot \nabla \mathbf{u}^* - \nu \Delta \mathbf{u}^* = 0 \qquad & \text{in } \Omega \\ \mathbf{u}^* = \mathbf{g} \qquad & \text{on } \Gamma. \end{aligned} \tag{4}$$

Then we obtain for the pressure

$$\frac{1}{\tau}(\mathbf{u}^{n+1} - \mathbf{u}^*) = -\nabla p \qquad \text{in } \Omega. \tag{5}$$

Taking the divergence of (5) yields

$$\operatorname{div} \mathbf{u}^{n+1} - \operatorname{div} \mathbf{u}^* = -\tau \Delta p \qquad \text{in } \Omega.$$

To satisfy equation (3b) we set $\operatorname{div} \mathbf{u}^{n+1} = 0$ and obtain the pressure correction equation

$$\begin{aligned} \Delta p = \frac{1}{\tau} \operatorname{div} \mathbf{u}^* \qquad & \text{in } \Omega \\ \nabla p \cdot \mathbf{n} = 0 \qquad & \text{on } \Gamma. \end{aligned}$$

Finally, we derive $\mathbf{u}^{n+1}$ by using the pressure p in (5) and obtain

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \tau \nabla p.$$

This method is known as *Chorin's projection method* [1]. Note that the pressure correction equation is ill-posed. To get a well-posed system we add a regularisation, e.g. $\int_\Omega p \, dx = 0$. Note that it is possible to decouple the convection and diffusion equation in (4) which yields the numerical scheme shown in algorithm 3.1.

Solve convection by moving point set

$$\vec{X}^{(n+1)} = \vec{X}^{(n)} + \tau \vec{\mathbf{u}}^{(n)}$$

Solve velocity equation

$$\tfrac{1}{\tau}(\vec{\mathbf{u}}^{(n+\frac{1}{2})} - \vec{\mathbf{u}}^{(n)}) = \Delta_X^h \vec{\mathbf{u}}^{(n+\frac{1}{2})}$$

Solve pressure equation

$$\Delta_X^h \vec{p}^{(n+1)} = \tfrac{1}{\tau} \nabla_X^h \cdot \vec{\mathbf{u}}^{(n+\frac{1}{2})}$$

Correct velocity

$$\vec{\mathbf{u}}^{(n+1)} = \vec{\mathbf{u}}^{(n+\frac{1}{2})} - \tau \nabla_X^h \vec{p}^{(n+1)}$$

Δ_X^h and ∇_X^h denote the discrete (FPM) counterparts to Δ and ∇ w.r.t. the point set $\vec{X}^{(n+1)}$. The above equations are equipped with the corresponding boundary conditions.

Algorithm 3.1: One time step of Chorin's projection with FPM. The point set is denoted by $(\vec{X}, \vec{\mathbf{u}}, \vec{p}) = \{(X_i, \mathbf{u}_i, p_i)\}_i$. Here X_i denotes the position of particle i and $\mathbf{u}_i$ and p_i the corresponding velocity and pressure.

4 Coupling Method

In this section we show the numerical implementation of the coupling of the fluid and the particle phase. We initialise the domain with two point sets. One is used for solving the fluid phase with smoothing length h_{fluid}, the other represents the initial distribution of the particles with a average distance of h_{part}. Note that h_{fluid} is fixed during the simulation whereas h_{part} changes, i.e. we obtain regions without particles and regions with a very dense distribution.

4.1 Mapping of Fluid Data to the Particle Phase

In a region we have the two possibilities $h_{fluid} < h_{part}$ and $h_{fluid} > h_{part}$. In the first case, we have more calculation points for the fluid phase than particles in a region as illustrated in figure 6 where the red (large) points represent the particle phase and the blue (small) ones the fluid phase. Here, the most efficient way is the mapping of the fluid phase data to each particle by deriving the interpolation coefficients at the position $\mathbf{x}_{part}$ as described in section 3.1. The fluid points needed for the approximation lie in the inner circle (figure 6).

If we have much more particles than fluid points, illustrated by figure 6 where now red is the fluid phase and blue the particle phase, this method would not be very efficient as we derive for each particle the approximation coefficients. Here, it is much more efficient to calculate an approximation polynomial using the fluid points marked by the outer circle which is used to approximate the particle quantities. In particular, this polynomial is, roughly speaking, derived by $f(x) := ax^2 + bx + c$ with the conditions $f(x_i) = f_i$ for all

fluid points i. Since we have more equations than unknowns, we obtain an overdetermined system. The resulting function f is used to map the desired quantity to the particle phase by $f_{part} := f(x_{part})$.

4.2 Numerical Implementation of the Coupling

The numerical implementation of the time step $n+1$ is organised as follows.

1. The fluid phase is solved for the velocity field $\mathbf{v}_{fluid}^{(n+1)}$ and the pressure $p_{fluid}^{(n+1)}$.
2. The values of the fluid phase are mapped to the particle phase, cf. section 4.1.
3. The acting forces are added by setting $\mathbf{v}_{part}^{(n+1)} = \mathbf{v}_{part}^{(n)} + \tau \mathbf{F}(\mathbf{v}_{fluid}^{(n+1)}, p_{fluid}^{(n+1)}, \ldots)$, where τ denotes the time step and $\mathbf{F}$ a function describing the particle force.
4. All particle-wall contacts are resolved according in section 2.2. In particular, each particle is checked for a wall contact in the current time step by neglecting all other particles, that is, no particle-particle contact is taken into account. If a contact occurs it is resolved and this particle will no longer be considered.
5. The remaining particles are checked for particle-particle interactions. If a contact occurs, the two particles are moved to the contact position, then the contact is resolved and finally, a particle-wall contact detection is performed. All particles with contact are moved to their final position for the current time step and will no longer be checked for interactions.
6. All remaining particles, which did not have a contact, are moved to their final position according to $\mathbf{v}_{part}^{(n+1)}$.

Especially for large time steps it is not possible to resolve all contacts and we allow a few particles to fly through each other. Since we only want to consider small particles later on, which also guarantees an appropriate particle distance, this is an acceptable behaviour. If too many particle contacts cannot be resolved, the time step has to be reduced. Note that resolving all contacts for a huge amount of particles in the system would be very expensive but not necessary to obtain a qualitatively good result for the particle phase.

4.3 Handling of the Total Kinetic Energy

The model of the particle-particle and -wall interaction also includes the coefficient of restitution. Hence, the total kinetic energy $E_{kin} = \frac{1}{2}\sum_{i=1}^{2} m_i|\mathbf{v}_i|^2 + I_i|\boldsymbol{\omega}_i|^2$ before the collision can be larger than the post-collision one. This difference is used to define an inner energy e_i for each particle. If this inner energy is too large, i.e. greater than a given threshold, the particle breaks into two parts. The breakage is modelled as a post-processing step. We reduce the radius and mass of the particle i by $r_i^{(n+1)} = \sqrt[3]{\frac{1}{2}} r_i^{(n)}$ and $m_i^{(n+1)} = \frac{1}{2} m_i^{(n)}$. Then a new particle j is generated by copying particle i. Finally, we rotate the velocity and angular velocity vector by random angles α, β and γ about the

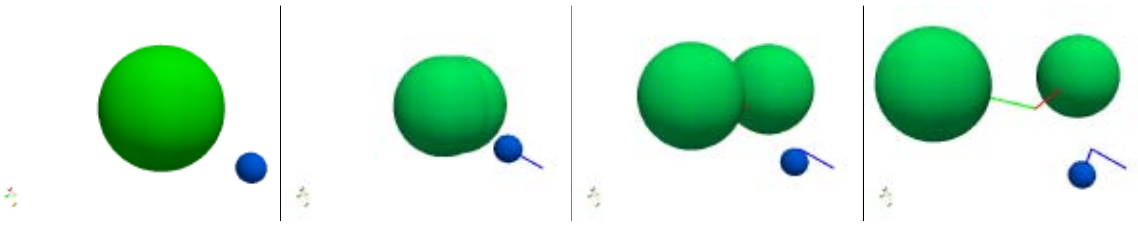


Figure 7: Breakage of a particle. In the beginning, only the blue particle moves. After the collision, the inner energy of the green ball gets large enough for a breakage. The lines represent the traces.

x-, y- and z-axis, respectively. Note that the angles should be greater than 0° and much less than 90°. Hence, we obtain

$$\mathbf{v}_i^{(n+1)} = \mathbf{R}(\alpha,\beta,\gamma)\mathbf{v}_i^{(n)} \qquad \mathbf{v}_j^{(n+1)} = \mathbf{R}(-\alpha,-\beta,-\gamma)\mathbf{v}_i^{(n)}$$
$$\boldsymbol{\omega}_i^{(n+1)} = \mathbf{R}(\alpha,\beta,\gamma)\boldsymbol{\omega}_i^{(n)} \qquad \boldsymbol{\omega}_j^{(n+1)} = \mathbf{R}(-\alpha,-\beta,-\gamma)\boldsymbol{\omega}_i^{(n)}$$

where $\mathbf{R}$ denotes the rotation matrix. Note that this is just a rough approximation but conserves the total kinetic energy. An example of applying this scheme is shown in figure 7.

Due to numerical issues also an increase of the total kinetic energy can occur which would increase rapidly over time. To avoid this case, E_{kin} is checked after each collision. If it increases, we scale the post-collision velocities and angular velocities until $E_{kin} < \bar{E}_{kin}$ where $\bar{E}_{kin}$ denotes the pre-collision energy.

5 Numerical Results

The model we have introduced is tested for a simple 2D particle separator, which separates light particles from heavy ones by a fluid flow. The schematic domain is shown in figure 8. The fluid inflow is on the left with an outlet on the right. The remaining boundaries are walls. Every $0.1s$ particles are injected at the top of the geometry which can leave the separator to the right and the bottom wall. For testing, the injected particles have variable mass which increases from the left to the right. Moreover, gravity is acting on the particles.

The results are shown in figure 9. The fluid flow is represented by streamlines. The heavy particles located at the right are mostly driven by gravity and not affected by the fluid flow strongly. In contrast, the light particles follow the fluid flow quickly until a collision occurs. The simulation shows that most light particles leave the separator to the right but there is also a certain amount of particles leaving the domain at the bottom due to collision with heavier particles.

6 Conclusion

For most applications the presented algorithm is sufficient and yields qualitatively good results. Since it does not resolve all particle-particle contacts, a huge amount of expensive

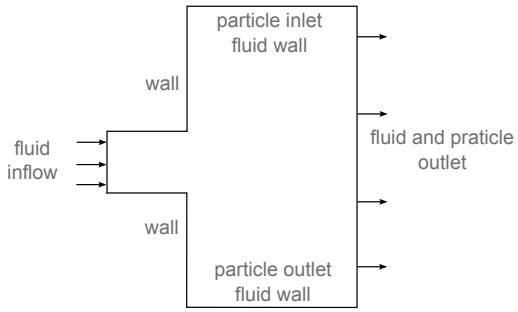


Figure 8: Schematic domain of a particle separator.

tests can be neglected which yields a very efficient algorithm for solving fluid-particle interactions. Also the presented mapping method increases the speed of the algorithm if we consider much more particles than calculation points for the fluid. This occurs for example in the simulation of dust in a gas flow. Moreover, the possibility of considering breakage is very useful for crushing processes.

REFERENCES

[1] A. Chorin. Numerical solution of the Navier-Stokes equations. *J. Math. Comp.*, 22:745–762, 1968.

[2] C.T. Crowe, M. Sommerfeld, and Y. Tsuji. *Multiphase flows with droplets and particles*. CRC, 1998.

[3] J. Kuhnert. *General smoothed particle hydrodynamics*. PhD thesis, University of Kaiserslautern, 1999.

[4] J. Kuhnert. Finite pointset method based on the projection method for simulations of the incompressible Navier-Stokes equations . *Springer LNCSE: Meshfree methods for Partial Differential Equations*, 26:243–324, 2002.

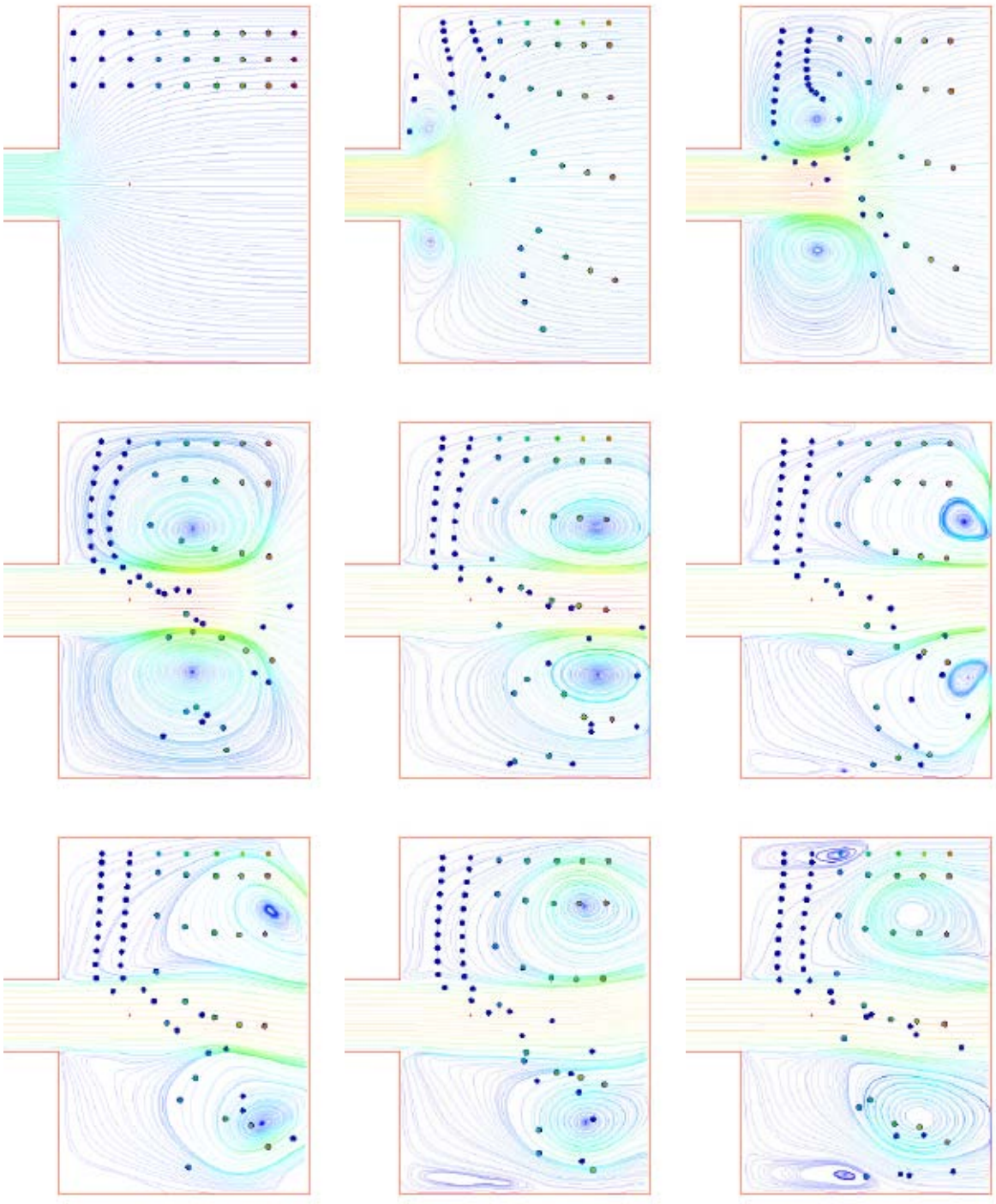


Figure 9: Simulation of a particle separator. Particles, whose weight increases from the left to the right, enter the device from above every $0.1s$. The background lines show the streamlines of the fluid flow. The time series goes from the left to the right for $t = 0, \ldots, 4$.